AD-A092 517   AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH   F/G 17/2
RECOVERY FROM NODE FAILURE IN A CONTRACT NET.(U)
AUG 80 P S PRINCE
UNCLASSIFIED   AFIT-CI-80-47T   NL

1 OF 1
AD A
092517

END
DATE
FILMED
DTIC

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LEVEL II

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT-CI-80-47T	2. GOVT ACCESSION NO. AD-A092	3. RECIPIENT'S CATALOG NUMBER 517
4. TITLE (and Subtitle) Recovery from Node Failure in a Contract NET.		5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Philip Steven Prince, Jr		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Massachusetts Institute of Technology		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE Aug 1980
		13. NUMBER OF PAGES 62
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC ELECTE DEC 0 5 1980 S D E

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs

Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A092517

DDC FILE COPY

012200

80-47T

RECOVERY FROM NODE FAILURE IN A CONTRACT NET

by

Philip Steven Prince Jr.

Accession For	
NTIS GRA&I	X
DDC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist.	Avail and/or special
A	

Submitted in Partial Fulfillment

of the Requirements for the

Degree of Bachelor of Science

at the

Massachusetts Institute of Technology

August, 1980

Signature of Author..
Department of Electrical Engineering, August 11, 1980

Certified by...
Thesis Supervisor

Accepted by..
Chairman, Departmental Committee of Theses

90-47T

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (ATC). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

Research Title: Recovery from Node Failure in a Contract NET

Author: Philip Steven Prince, Jr.

Research Assessment Questions:

1. Did this research contribute to a current Air Force project?

a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

a. Man-years ______. b. $________

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3 above), what is your estimate of its significance?

2. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the back of this questionnaire for your statement(s).

NAME GRADE POSITION

ORGANIZATION LOCATION

USAF SCN 75-20B

TABLE OF CONTENTS

Section		Page
1.	Description of STRIPS	2
2.	Description of contract net	5
3.	Contract net distribution of STRIPS	11
4.	Detailed example of distributed STRIPS	20
5.	Description of problem of failed node	35
6.	Methods considered for reccvery	37
7.	Example of recovery	52
8.	Summary	56
9.	History of problem	58
10.	Suggestions for future work	59
11.	Footnotes	61
12.	References	62

LIST OF ILLUSTRATIONS

Figure		Page
1.	Example of search tree	3
2.	Task announcement format	6
3.	Bid format	7
4.	Award format	8
5.	Report message	8
6.	Contract message protocol	9
7.	Task announcement example	10
8.	Bid example	12
9.	Award example	12
10.	Report example	13
11.	Distributed STRIPS and/or goal tree	17
12.	Picture of example problem	20
13.	K-tary goal tree	35
14.	Status request	39
15.	Status reply	40
16.	Acknowledgement	40
17.	Status report	42
18.	Recontract announcement	44
19.	Recovery from failed node	45
20.	Cost of recovery methods	49

ABSTRACT

The problem of a catastrophic node failure in a contract net will be addressed. STRIPS will be distributed by means of the contract net protocol, and used as a medium for studying the methods that might be used to recover from the node failure.

Four methods will be described, and compared, with respect to normal and recovery operational costs.

This paper was completed under the supervision of Randall Davis, Assistant Professor, Electrical Engineering and Computer Science.

1. DISCRIPTION OF STRIPS

The Stanford Research Institute Problem Solver, STRIPS,[1] has been included in this paper as a medium for studying methods of recovering from catastrophic failure of nodes in a contract net.

STRIPS uses a theorem resolution mechanism to determine differences between a current state and a goal state and uses it to verify correct means of transferring from one state to the other. STRIPS also has a list of operators, with their relevant preconditions and effects, which can change the states.

Here is a brief discription of how a problem is resolved. A present world model M_0 and a goal state G_0 is made out of well-formulated formulae WFF's. The theorem prover is asked to show that G_0 is included in M_0. If it is successful then the problem is solved, if it is not the incomplete proof is used as the difference between M_0 and G_0. STRIPS then goes to its collection of operators to see if any are relevant in reducing this difference. Of these operators having the desired effect, the operator O, which has the fewest literals in its preconditions, is chosen as the best to try first. A new world model M_1 is constructed by applying the operator to the old world model, $M_1 \longleftarrow OM_0$. STRIPS is called recursively, with the problem now to see if G_0 is included in M_1. Termination occurs when a world model is found that includes the goal state and the operators are returned in proper order.

There are two issues which should be discussed further. There may be several "best" operators which can reduce the difference. The operators, themselves, have preconditions which may not be immediately met in M_o.

When there is a tie between several operators STRIPS arbitrarily picks one. Should this choice prove to be fruitless, STRIPS backtracks and tries the next best choice. It fails to solve a problem if no operators can be found to reduce the differences.

The preconditions of the operators must be met for an operator to be applied. STRIPS handles this by setting up the preconditions as a goal state and trying to reduce the difference between this goal state and the current world model.

The following is a clearer discription of the problem solving process.

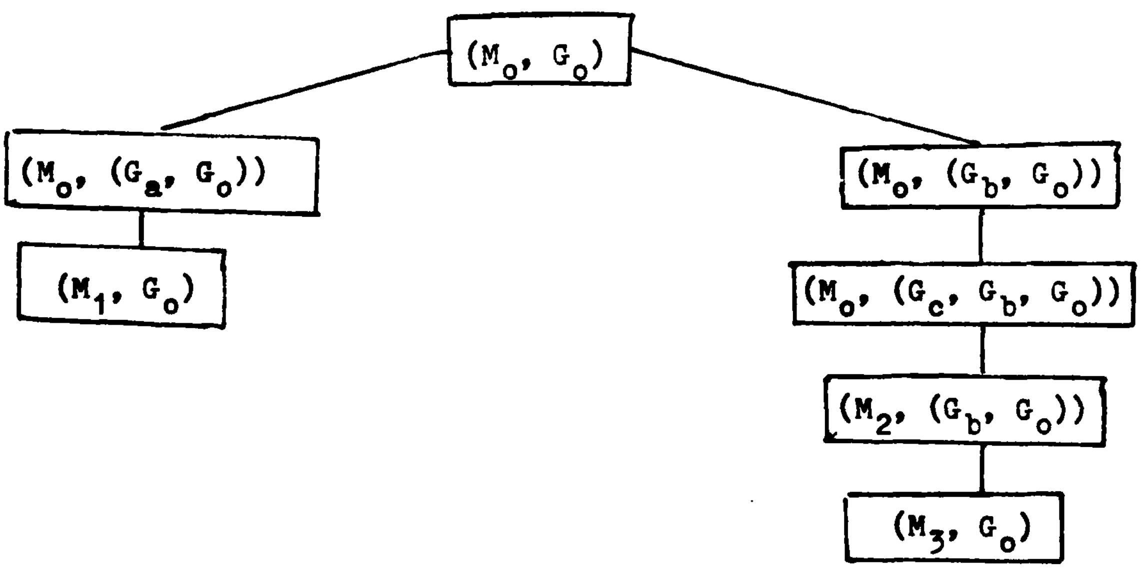


Figure 1. Example of Search Tree

M_o and G_o are formulated. The theorem prover is asked to show that G_o is included in M_o. When it is unable to do so, the incomplete proof is used as the difference D_o between M_o and G_o. STRIPS then goes to its pool of operators to find those which can reduce D_o, and finds that O_a and O_b are relevant to the problem, with O_a being the better choice.

The preconditions for O_a are set up as a subgoal G_a, which is placed before G_o. The theorem prover is asked if M_o includes G_a. In this case it is, so O_a is applied to M_o forming M_1. G_a is removed from the goal list and STRIPS is called on (M_1, G_o). After determining the difference between M_1 and G_o, it is found that no operators are relevant, so O_b is then tried. Once again a subgoal, G_b, is constructed from the preconditions of the operator O_b and is placed before the final goal state. STRIPS finds that G_b is not included in M_o and that the difference between them can be reduced by operator O_c. The subgoal G_c is constructed from the preconditions of O_c and is placed before G_b in the goal list. It is found that G_c is included in M_o, so O_c is applied to M_o forming M_2. M_2 includes G_b, so M_3 is formed. At the last node the theorem prover finds that G_o is included in M_3, therefore the process terminates with O_c O_b as the correct operator sequence. STRIPS uses best-first search to determine which node to expand.

2. DESCRIPTION OF CONTRACT NET

STRIPS lends itself to distribution and parallelism in several areas.

(1) The differences between states could possibly be handled more efficiently if processors could specialize in a class of differences.

(2) Parallelism can be introduced to simultaneously look into possibilities generated by having several relevant operators available.

(3) Once an operator O_i has been selected there are two subproblems, (M_{i-1}, G_i) (M_i, G_{i+1}), which may be handled concurrently.

In order to achieve these possible gains I will use a contract net[2].

A contract net is a set of nodes or processors that are able to communicate with each other. Communication is achieved by exchanging messages which certain information relevent to particular tasks they are showing, and it is standardized by means of a protocol.

When a node has a task that it wishes to have some assistance in completing, it issues a *task announcement*. A task announcement can be directed to any subset of the nodes in the net.

The task announcement has a header, which contains the sender, intended recipient of the message, message type, and a contract number. It also has slots, which contain a discription of the task, requirements a node must have to

perform the task, information it wishes to know about a node, and an expiration time. Communication is considered to be a valuable resource so the information in these slots is given by keywords and abbreviations known by the nodes.

To

From

Type

Contract

Task Abstraction

Eligibility Specification

Bid Specification

Expiration Time

Figure 2. Task Announcement Format

A node, which thinks it can perform the task in the task abstraction, and which meets the eligibility specifications answers the announcement with a bid.

To

From

Type

Contract

Node Abstraction

Figure 3. Bid Format

The node abstraction slot contains the information required by the announcing node in the bid specification slot. This information will be used by the announcing node as criteria for selecting which node it wishes to share the task with. When the announcing node has selected who it wants, it sends an award message to that node. The award message has a task specification slot which contains all relevant information necessary for completion of the task.

This exchange of task announcement, bid and award messages is the protocol for establishing a contract. The announcing node is the manager of the task and the bidding node is the contractor of the task. The contractor can then play the role of the manager itself, by contracting out portions of its task.

To

From

Type

Contract

Task Specification

Figure 4. Award Format

When a task has been completed by a node, it sends its results to the manager in a report message. The report contains a result description slot that specifies the results of the execution.

To

From

Type

Contract

Result Description

Figure 5. Report Message

A report message can either be an interim report or a final report type depending upon at what stage of completion it is issued.

The manager also has the option of terminating a contract before it has been completed by means of a termination message. When a contractor receives this message it stops working on that contract and cancels all related subcontracts.

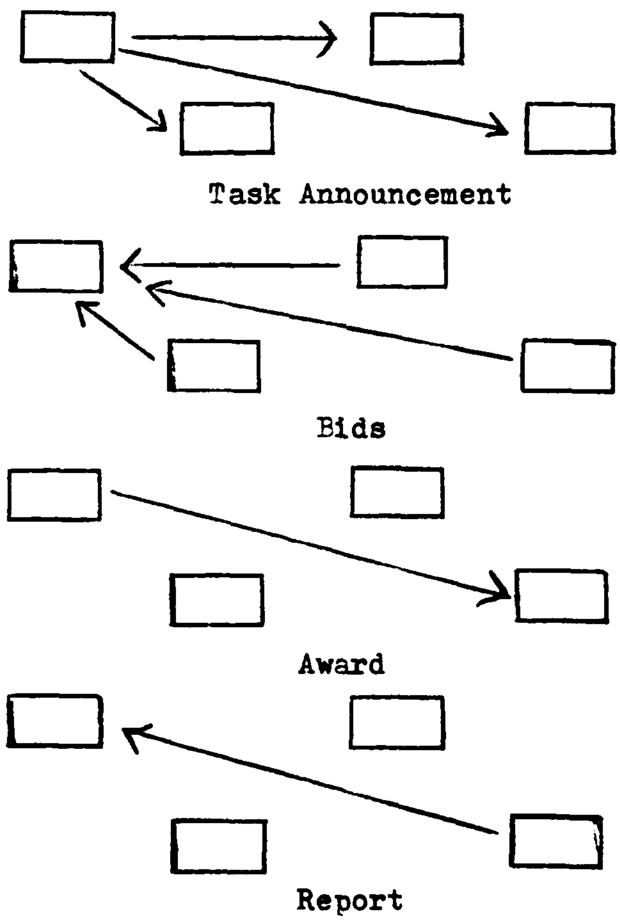


Figure 6. Contract Message Protocol

3. CONTRACT NET DISTRIBUTION OF STRIPS

The nodes must have a common theorem prover and a list of relevant operators to the task. The theorem will be required in the eligibility specification of a task announcement. The operators themselves will be contained in the nodes so that when a node receives an announcement it can determine if it has any relevant operators for reducing the differences in the task announcement. As an abstraction of the task, the difference D_0 between M_0 and G_0 will be included in the task abstraction slot of the task announcement. The bid specification will require that bids contain, in their node abstractions, when the node will be available, the number of literals in the preconditioned clauses of the relevant operator, and the particular instantiation of that operator that will be used. The award message will contain in its task specification slot the current world model M_0 and the goal state G_0. When the contractor reports, the result description slot will contain the list of operators that reduce the difference in the order of application.

To: *

From: NODE 1

Type: TASK ANNOUNCEMENT

Contract: A1

Task Abstraction:

TASK TYPE: DIFFERENCE REDUCTION

ABSTRACTION: D_0

Eligibility Specification:

MUST HAVE: THEOREM-PROVER XX

Bid Specification:

WHEN AVAILABLE

NUMBER OF LITERALS

RELEVANT INSTANTIATION

Expiration Time:

12:04:36 2-6-79

Figure 7. Task Announcement Example

To: NODE 1

From: NODE 2

Type: BID

Contract: A1

Node Abstraction:

12:04:39 2-6-79

3

O_1 (A, B)

Figure 8. Bid Example

To: NODE 2

From: NODE 1

Type: AWARD

Contract: A1

Task Specification:

(M_i, G_{i+1})

Figure 9. Award Example

To: NODE 1

[illegible]

Type: FINAL REPORT

Contract: A1

Result Description:

(O_1, O_2, O_3)

Figure 10. Report Example

Before discussion on the problem solving process can continue some issues must be solved or clarified.

(1) What information is needed to allow a node to make an intelligent bid?

(2) What information is needed for the manager to intelligently distribute the task?

(3) When should a manager terminate a contract?

(4) Is it appropriate to distribute a task into two subtasks, where the first examines the differences between a goal state and an intermediate goal, and the second examines the differences between a world model, formed by the operator related to that intermediate goal state, and the next goal state?

The task abstraction contains the information perti-

nent to a bidder selecting a task. The task name gives [illegible] the class of problems that he is expected to work on. The difference D_0 is the minimum information needed by the bidder to determine if his operators are relevent to the task. With this information a bidder can decide if it wants to do the task and determine what instantiation of its operators is appropriate.

The manager will use the information requested in the bid specification to evaluate the nodes usefulness. There are two issues here. What constitutes a more useful node? When should contracts be established with less useful nodes?

In STRIPS, the heuristic for determining the best node to expand was the one whose operator preconditions had the fewest literals. This criterion will be used again here. STRIPS, however, did not need to know the name of the operator since it only had one list of operators to choose from. The contract net may have many nodes with the same capabilities that will give bids having the same number of literals. It would be inappropriate to contract all of them, since they would all be doing the same thing. The particular operator instantiation is therefore required to allow the manager to refrain from this excessive redundancy. The when available information allows a manager to break a tie between nodes having the same least number of literals, and the same particular operator instantiation. If two bids are still tied after all this, the manager

arbitrarily picks one.

The problems handled by STRIPS are nondeterministic in that a best choice at a particular level of search may not provide the answer. Therefore, it is necessary to keep track of all possible courses of action and it may be useful to expand and thus contract with less useful nodes. A problem arises here. Some tasks may have an infinite number of solutions. If a manager waits for all of these to come back it will be waiting a very long time.

In this paper, it is assumed that there are ample nodes available, so a manager will be allowed to contract with the most useful nodes and all the alternatives. However, when the manager receives a solution to the task from one of its contractors, it will terminate the remaining, outstanding contracts. This is an adequate approach because it will be assumed, for this paper, that a node reporting before another has the better solution.

This distributed STRIPS establishes an and/or goal tree. In addition to terminating as discussed above, a manager will want to terminate contracts to members of an and branch when one of the nodes reports that it has failed to accomplish its contract.

The world models, goal states and operators that will be used as examples later, are simple enough so that an operator can be applied to a world model, without its preconditions being met. This facilitates the added parallelism of breaking a task into concurrent subtasks. This is not necessarily true for all problems that can be

handled by STRIPS.

A generalized example to illustrate the operation of the process, follows.

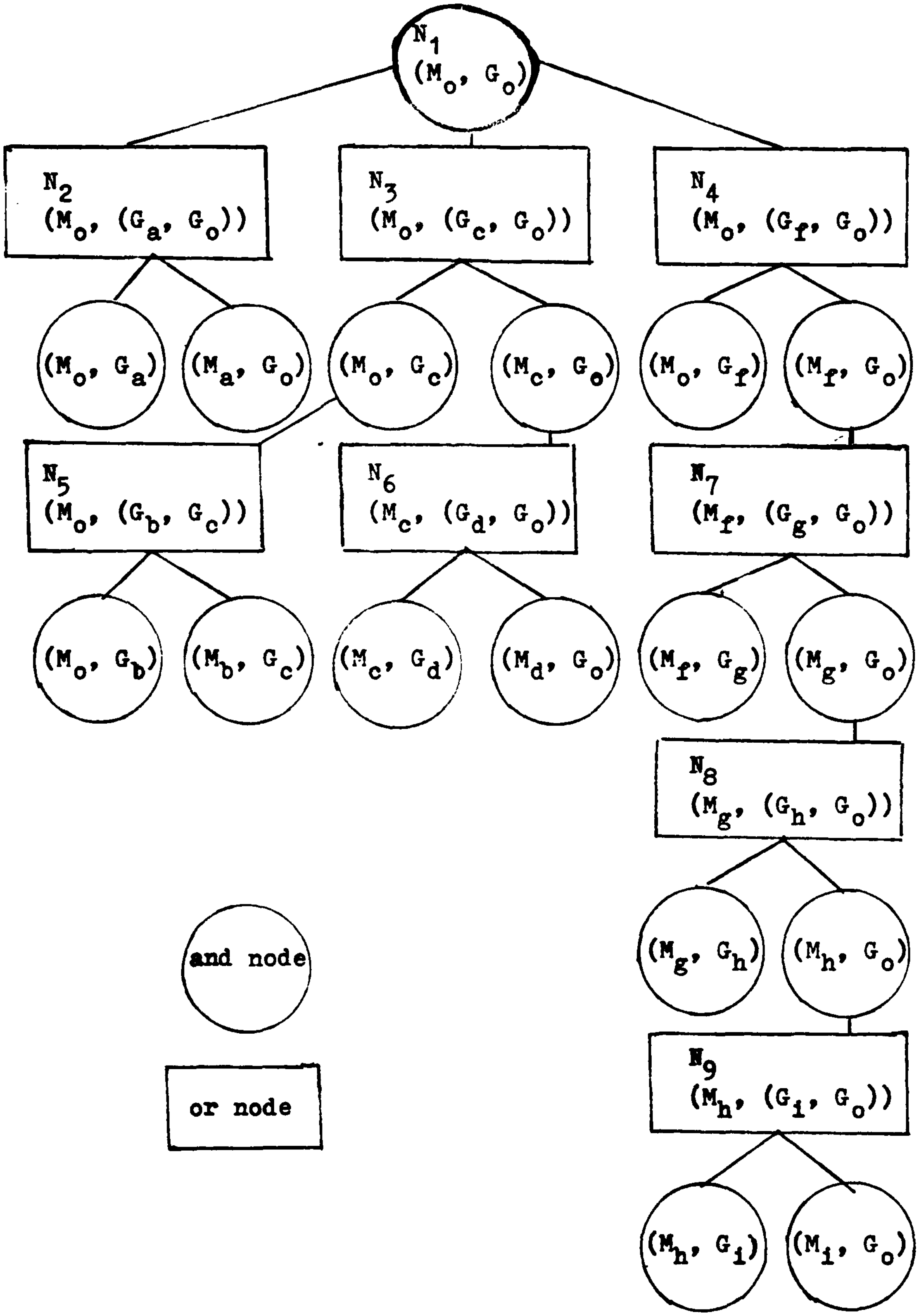


Figure 11. Distributed STRIPS and/or Goal Tree

The problem is reducing the difference between M_o and G_o. The operators O_a, O_c, and O_f are equally relevant at the first level. O_a is a dead end and will not result in a solution. O_c is included in the solution O_b, O_c, O_d. O_f is the first operation in the solution O_f, O_g, O_h, O_i.

N_1 receives the original task to reduce the difference between M_o and G_o. It computes the difference D_o between them and announces the task to reduce D_o. Nodes N_2, N_3, and N_4 examine D_o and see that they have operators O_a, O_c, and O_f that are relevant. N_1 contracts with them to reduce a difference using their respective operators. N_2 establishes goal G_a from the preconditions from O_c. It computes the difference between N_o and G_a and announces the task of reducing it. N_2 then constructs M_a by applying O_a to M_o and announces the task of reducing the difference between them. In this example no nodes can reduce either of the differences so N_2 reports to N_1 that it has failed. If N_1 had any other contracts which depended upon N_2's results it would now terminate them. N_3 establishes G_c from the preconditions of O_c. It computes the difference between M_o and G_c and announces the task of reducing that difference. N_5 sees that it has operator O_b relevant to the task. After N_5 contracts with N_3 to reduce that difference, it establishes G_b from the preconditions of O_b. N_5 determines that there is no difference between M_o and G_o. It then computes M_b by applying O_b to M_c and finds there is no difference between M_b and G_c. Its subtasks

completed, N_5 reports to N_3 *giving* as its result O_b. While N_5 had been working, N_6 had been working on a similar problem, announced by N_3 after it had computed M_c by applying O_c to M_o. N_6, working in the same manner as N_5, has determined that O_d is a good operator because there is no difference between M_c, G_d, and between M_d, G_o so it reports to N_3 operator O_d. N_3 receiving operators O_b from N_5, and O_d from N_6 combines them with O_c and reports to N_1 that (O_b, O_c, O_d) satisfies its contract. While N_3 had been laboring on its own contract. The process below N_4 operates in the same manner as the previously discussed nodes, with all of the leaf nodes, in its branch of the tree, having no difference between their respective current world model and goal state. It reports to N_1 that (O_f, O_g, O_h, O_i) satisfies its contract. It is important to know what happens if N_3 or N_4 reports before the other. If N_3 reports first, N_1 notifies N_4 that it wishes to terminate its contract. N_4, in turn, notifies N_7 that it wishes to terminate its contract, and so on down the line. It has been assumed that all processors work at the same rate, so the best solution has been picked in this case. Clearly, if processors don't work at the same rate, some work will have to be done to determine the optimum solution.

The search pattern has now been changed from the best-first search employed by STRIPS to a breadth-first search. The optimum solution in this case has been found in the time it would take STRIPS to find it, if STRIPS made all the right choices.

4. DETAILED EXAMPLE OF DISTRIBUTED STRIPS

The example used by Fikes and Nilsson[3] to illustrate the operation of STRIPS will be used to give a more detailed illustration of distributed STRIPS.

The problem is to determine a plan which a robot, in a room with three boxes, could use to arrange the boxes so that they would be at the same location.

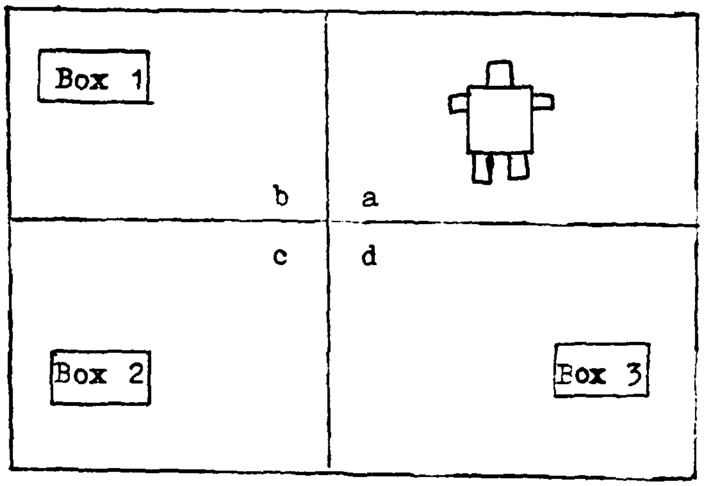


Figure 12. Picture of Example Problem

The clauses used to describe the world model simply give the positions of the robot and the three boxes.

(1) ATR(x): The robot is at location x.

(2) AT(y,z): The object y is at location z.

There are two operators which can change the world model. An operator makes the change by deleting the clauses from the world model that are contained in the operators delete list, and by adding clauses that are in its add list.

(1) push (k, m, n): Robot pushes object k from location m to location n.

Precondition: AT(k, m) $\wedge$ ATR(m)

Delete List: ATR(p) AT(k, p)

Add List: AT(k, n) ATR(n)

(2) goto (m, n): Robot goes from location m to location n.

Precondition: ATR(m)

Delete List: ATR(p)

Add List: ATR(n)

The delete lists have been modified with respect to the original example. The variable p is just used as a place holder. When AT(r, p) or ATR(p) is deleted from the world model, the location of the robot or object is irrevelant to the deletion. Applying goto (j, b) to a world model will result in the robot being placed at b irregardless of the value of j.

The purpose of this modification is to facilitate current reduction of the difference between the current world model and operators preconditions, and the difference between the new world model, formed by applying the operator and the goal state.

The initial world model of this task is given by

Mo: ATR(a)

AT(Box 1, b)

AT(Box 2, c)

AT(Box 3, d)

The goal of the task is described by

$$G_0: \; (\exists x)\,(AT\,(Box\ 1,\ x) \wedge$$
$$AT\,(Box\ 2,\ x) \wedge$$
$$AT\,(Box\ 3,\ x).$$

This problem is interesting as an example because there are an infinite number of possible solutions and because there are many solutions with the same, even the optimum, level of complexity.

STRIPS uses the QA 3.5 theorem prover[4] to determine the differences between states. Only those differences, not the computation of them, will be repeated here.

The first node, Node 1, which assumes the role of the overall task manager, determines that the difference between M_0 and G_0 is

$$\sim AT\,(Box\ 1,\ c) \vee \sim AT\,(Box\ 3,\ c)$$
$$\sim AT\,(Box\ 2,\ b) \vee \sim AT\,(Box\ 3,\ b)$$
$$\sim AT\,(Box\ 1,\ d) \vee \sim AT\,(Box\ 2,\ d)$$

Node 1 needs to have this difference reduced so it issues a task announcement.

To : *

From: NODE 1

Type: TASK ANNOUNCEMENT

Contract: 1

Task Abstraction:

TASK TYPE: DIFFERENCE REDUCTION

ABSTRACTION: $\sim AT\,(Box\ 1,\ c) \vee \sim AT\,(Box\ 3,\ c)$

$\sim AT\,(Box\ 2,\ b) \vee \sim AT\,(Box\ 3,\ b)$

$\sim$ AT (Box 1, d) $\vee \sim$ AT (Box 2, d)

Eligibility Specification:

MUST HAVE: THEOREM PROVER QA3.5

Bid Specification:

WHEN AVAILABLE

NUMBER OF LITERALS

RELEVANT INSTANTIATION

Expiration Time:

12:04:36 2-6-79

The only relevant operator is push (k, m, n) but there are six relevant instantiations push (Box 1, m, c), push (Box 1, m, d), push (Box 2, m, b), push (Box 2, m, d), push (Box 3, b) and push (Box 3, c).

An operator having several relevant instantiations is a difficult issue to resolve. It will be assumed that enough nodes will bid to allow Node 1 to contract each instantiation with a different node. If not, Node 1 will be required to remember uncontracted instantiations, and the node that offered them, should the selected instantiations fail.

The operation following each node below Node 1 will be identical, except for different variables used as parameters to the different instantiations. Expanding one will be representative of them all.

To: NODE 1

From: NODE 2

Type: BID

Contract: 1

Node Abstraction:

12:05:06 2-6-79

(2, 2, 2, 2, 2, 2)

(push (Box 1, m, c), push (Box 1, m, d),
push (Box 3, m, b), push (Box 2, m, d),
push (Box 3, m, b), push (Box 3, m, c))

Node 2 now seals the contract with the award message.

To: NODE 2

From: NODE 1

Type: AWARD

Contract: 1-3

Task Specification:

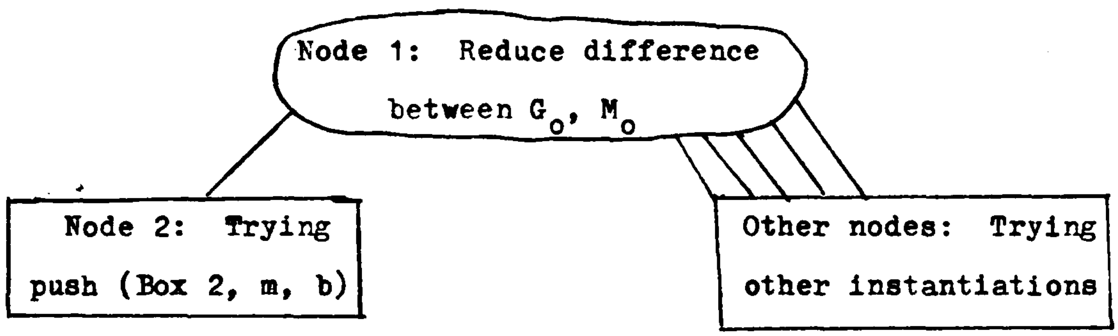
push (Box 2, m, b)

((ATR (a) AT (Box 1, b)
AT (Box 2, c) AT (Box 3, d)),
(($\exists$x) (AT (Box 1, x) $\wedge$
AT (Box 2, x) $\wedge$
AT (Box 3, x)))

The instantiation push (Box 2, m, b) has been included in the task specification so that Node 2 will know which

instantiation offered it is to use. The -3 has been appended to the contract number so that Node 1 can keep track of which instantiation it gave to Node 2.

Expansion of goal tree so far.



When Node 2 receives the award it uses AT (Box 2, c) from M_o to refine the operator instantiation to push (Box 2, c, b). It breaks the task into two subtasks. The first is to see if the operator preconditions are met in M_o. The second is to see if G_o is included in the world model formed by applying push (Box 2, c,b) to M_o.

Node 2 determines that the difference between M_o and the precondition ATR(c). It issues a task announcement.

To: *

From: NODE2

Type: TASK ANNOUNCEMENT

Contract: 2

Task Abstraction:

TASK TYPE: DIFFERENCE REDUCTION

ABSTRACTION: ~ ATR(c)

Eligibility Specification:

MOST HAVE: THEOREM PROVER QA3.5

Bid Specification:

WHEN AVAILABLE

NUMBER OF LITERALS

RELEVANT INSTANTIATION

Expiration Time:

12:05:36 2-6-79

The only instantiation, goto(m, c), leads to an optimum solution.

To: NODE 2

From: NODE 3

Type: BID

Contract: 2

Node Abstraction:

12:06:06 2-6-79

2

goto(m, c)

Node 2 answers with

To: NODE 3

From: NODE 2

Type: AWARD

Contract: 2-1

Task Specification:

goto(m, c)

((ATR (a) AT (Box 1, b)

AT (Box 2, c) AT (Box 3, d)),

(ATR (m)))

Before announcing the second subtask, Node 2 forms another world model by applying push (Box 2, c, b) to it.

M_1: ATR (b)

AT (Box 1, b)

AT (Box 2, b)

AT (Box 3, d)

It finds the difference between M_1 and G_0 are

$\sim$ AT (Box 1, d) $\vee$ $\sim$ AT (Box 2, d)

$\sim$ AT (Box 3, b).

The only relevant operator is push (k, m, n) with three instantiations, push (Box 1, m, d) push (Box 2, m, d) and push (Box 3, m, b). The first two will lead to non-optimum solutions. Node 2 issues the task announcement.

To: *

From: NODE 2

Type: TASK ANNOUNCEMENT

Contract: 3

Task Abstraction:

TASK TYPE: DIFFERENCE REDUCTION

ABSTRACTION: $\sim$ AT (Box 3, b)

$\sim$ AT (Box 1, d) $\vee$ $\sim$ AT (Box 2, d)

Eligibility Specification:

MUST HAVE: THEOREM PROVER QA3.5

Bid Specification:

WHEN AVAILABLE

NUMBER OF LITERALS

RELEVANT INSTANTIATION

Expiration Time:

12:06:36 2-6-79

Node 4, among others, makes a bid.

To: NODE 2

From: NODE 4

Type: BID

Contract: 3

Node Abstraction:

12:07:06 2-6-79

(2, 2, 2)

(push (Box 1, m, d), push (Bid 2, m, d), push (Box 3, m, d))

Node 2 selects Node 4 to work on push (Box 3, m, b).

To: NODE 4

From: NODE 2

Type: AWARD

Contract: 3-3

Task Specification:

push (Box 3, m, b)

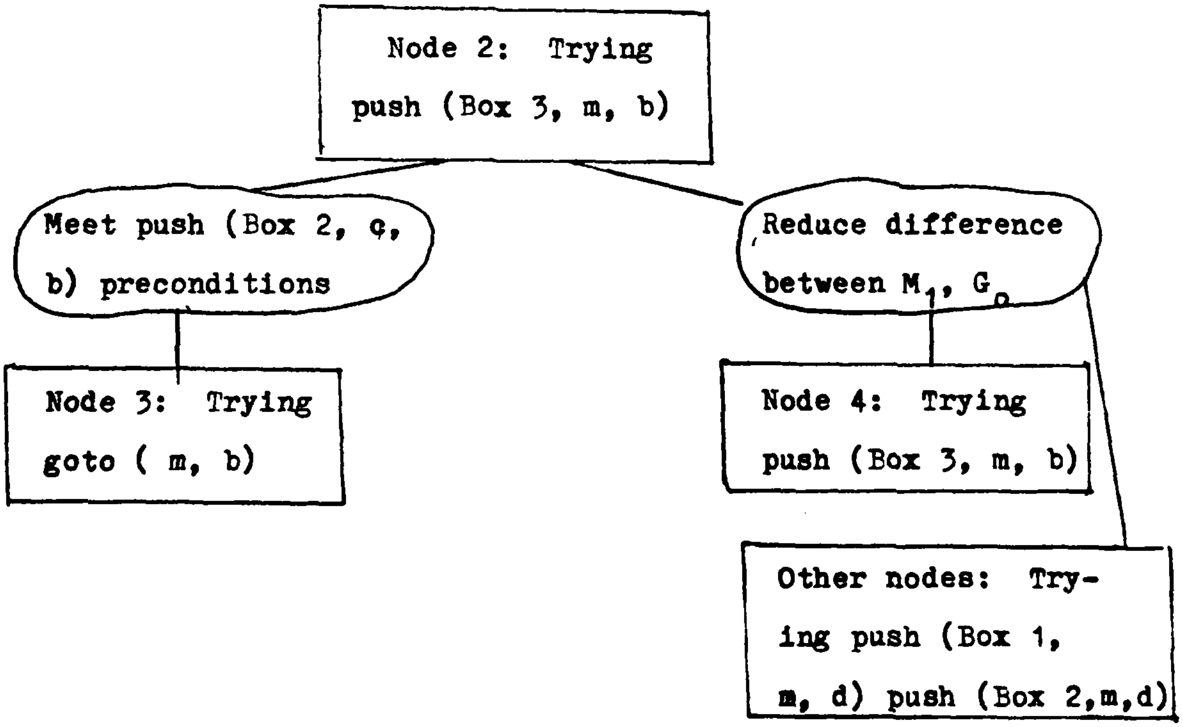
((ATR (b) AT (Box 1, b)

AT (Box 2, b) AT (Box 3, d)),

(($\exists$x) (AT (Box 1, x)$\wedge$ AT (Box 2, x)$\wedge$

AT (Box 3, x)))

Expansion of goal tree from Node 2 so far.



To: *

From: NODE 4

Type: TASK ANNOUNCEMENT

Contract: 4

Task Abstraction:

TASK TYPE: DIFFERENCE REDUCTION

ABSTRACTION: - ATR (d)

Bid Eligibility:

MUST HAVE: THEOREM PROVER QA3.5

Bid Specification:

WHEN AVAILABLE

NUMBER OF LITERALS

RELEVANT INSTANTIATION

Expiration Time:

12:07:36 2-6-79

It forms the new world model by applying push (Box 3, d, b) to it.

M_2: ATR (b) AT (Box 1, b)

AT (Box 2, b) AT (Box 3, b)

Node 4 finds no difference between M_2 and G_0, so it knows push (Box 3, d, b) is valid, provided the operators' preconditions are met.

There is one operator instantiation which will reduce

the difference of the node abstraction goto (m, d). Node 5 has goto (m, n) and bids on the task.

To: NODE 4

From: NODE 5

Type: BID

Contract: 4

Node Abstraction:

12:08:06 2-6-79

1

goto (m, d)

Node 4 awards the contract to look into goto (m, d) to Node 5.

To: NODE 4

From: NODE 5

Type: AWARD

Contract: 4-1

Task Specification:

goto (m, d)

((ATR (b) AT (Box 1, b)

AT (Box 2, b) AT (Box 3, d)),

(ATR (m)))

Node 3 uses ATR (a) to refine goto (m, b) to goto (a, b). It finds that the preconditions for goto (a, b) are met in G_0. It also finds the world model, formed by applying goto (a, b) to M_0 includes the goal state it received. Node 3 makes its report to Node 2.

To: NODE 2

From: NODE 3

Type: FINAL REPORT

Contract: 2-1

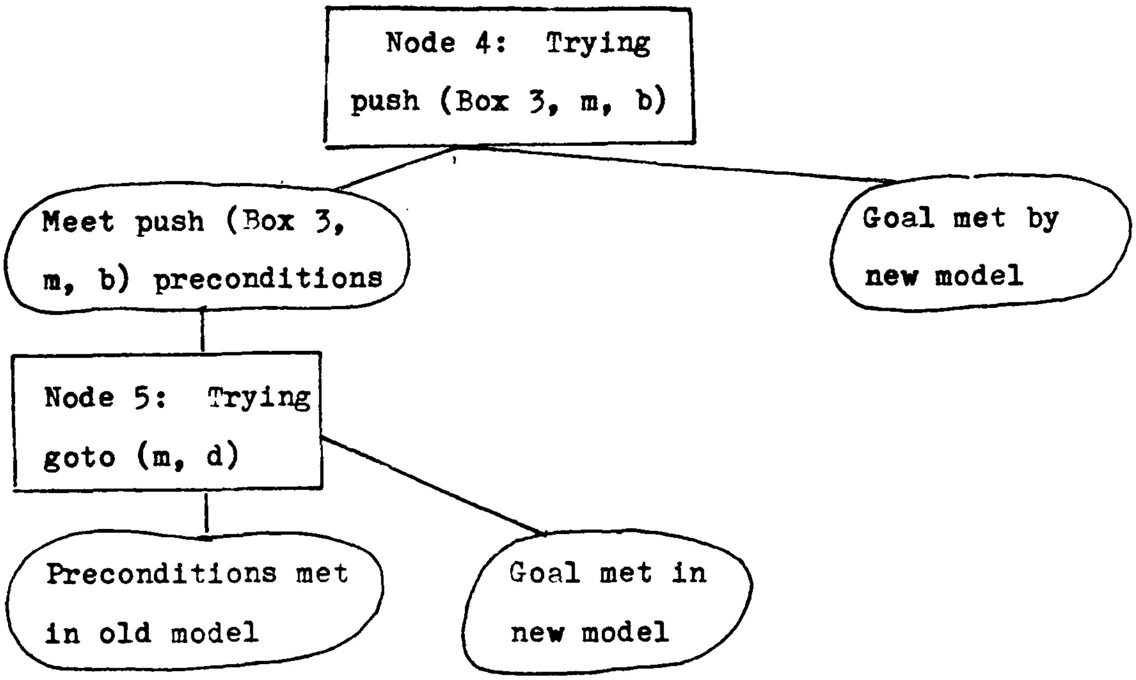
Result Description:

goto (a, b)

Node 2 waits for results from Node 4.

Node 4 uses AT (Box 3, d) to refine push (Box 3, m, b) to push (Box 3, d, b). It then determines that the differences between M_1 and the preconditions ATR (d) AT (Box 3, d) is - ATR (d). To reduce this, Node 4 makes a task announcement.

Further extension of goal tree.



Node 5 refines goto (m, d) to goto (b, d) to goto (b, d) by using ATR (b) from the old world model. It finds there is no difference between M_1 and the preconditions of goto (b, d). After applying goto (b, d) to M_1 to find a new world model, Node 5 finds there is no between this model and the goal state it received. It can now report to Node 4 that its result is goto (b, d)

To: NODE 4

From: NODE 5

Type: FINAL RESULT

Contract: 4-1

Result Description:

goto (b, d)

When Node 4 receives the report from Node 5 it prepares its report for Node 2. Combining goto (b, d) and push (Box 3, d, b) yeilds the result goto (b, d), push (Box 3, d, b).

To: NODE 2

From: NODE 4

Type: FINAL REPORT

Result description:

goto (b, d), push (Box 3, d, b)

When Node 2 receives the report from Node 4, it combines the result from Node 2 its operator, push (Box 2, c, b), and the results from Node 4.

To: NODE 1

From: NODE 2

Type: FINAL REPORT

Contract: 1-3

Result Description:

goto (a, c), push (Box 2, c, b), goto (b, d), push (Box 3, d, b)

Node 2 then terminates the contracts it has with the nodes working on push (Box 1, m, d) and push (Box 2, m, d).

Node 1 cancels its outstanding contracts, upon receipt of Node 2's report, by termination messages.

5. DESCRIPTION OF PROBLEM OF FAILED NODE

It is apparent now, that problem solving by a contract net distributed STRIPS can be represented by an N^{th} level 2-K and/or goal tree. Generalizing it for any hierarchical contract net yields an n^{th} level K-tary goal tree. Studying the nodal relationships on such a tree, will allow an understanding of the effects of a catastrophic node failure.

In the unmodified contract net a task employs K^n nodes and has a cost C (x) associated with the cost of computing, in achieving that task. The total cost for achieving a task will be the sum of the cost of computation and the cost associated with the messages. Letting TA be the cost of a task announcement, B be the cost of the bid, A be the cost of an award and R_p be the cost of a report, the total cost is $K^nTA+K^nB+K^nA+K^nR_p+C(x)$ or $K^n(TA+B+A+R_p)+C(x)$.

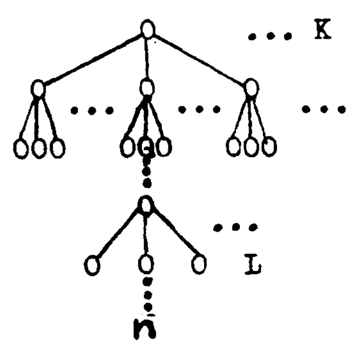


Figure 13. K-tary Goal Tree

A node, failing at level L, will result in the contracts for K^{n-L} nodes [illegible], thus causing the system to fail. It is very important to determine methods for recovery from such a catastrophe. This problem is further compounded if nodes are allowed to contract several tasks at a time. A single node failure could potentially cause widespread damage among unrelated tasks.

6. METHODS CONSIDERED FOR RECOVERY

Four methods will be considered here for recovery from nodal failure.

(1) R^{th} order redundancy

(2) Periodic status inquiries, coupled with re-announcing the tasks of the failed nodes.

(3) Periodic status inquiries, coupled with interim status reports to facilitate recovery of subcontracts.

(4) Continuous communication between manager and contractor.

The first method that comes to mind, that requires no additional message types, is to utilize R^{th} order redundancy. For every task, the manager would simply announce, accept bids for, and award R contracts. The number of messages in this approach then goes to $(RK)^n$ and the cost is $R^nK^n(TA+B+A+R_p)+R^nC(x)$, for those messages plus the cost of computation. If a node fails there are R-1 nodes still available to give the needed results. This is still an imperfect fix. In the event of a catastrophic failure, where all R nodes fail, there is no recourse for recovery available.

Since communication between nodes is only made at contract time or when results are reported, it is impossible for a manager to detect the failure of one of its contractors due to a catastrophe. This necessitates the introduction of status request messages. When a manager

has waited a predetermined time, without receiving a report from its contractor, it may issue a status request to its contractor. If the contractor is still functioning it will reply. When the manager receives the reply, it knows all is well and so continues to wait for a report. If it does not receive a reply, it reannounces the task that it had contracted to the failed node, and continues the achievement of its own tasks, through a new subcontractor.[4] The advantages of this approach are that it is simple and the status messages can easily be formed from existing message formats. The disadvantage of this approach is that all computation below the failed node is still lost to the manager, and that the nodes below that will continue work futilely.

A status request can be constructed from a task announcement by placing in the task abstraction slot the task name, status request, and by placing in the bid specification slot the required status. The expiration time slot will be coded for immediate response, indicating to the contractor, that when he receives this message, the manager expects him to reply before completing any more work.

To: CONTRACTING NODE

From: MANAGING NODE

Type: TA

Contract: CURRENT CONTRACT NUMBER

Task Abstraction:

TASK TYPE: STATUS REQUEST

Bid Specification:

STATUS

Expiration Time:

IMMEDIATE RESPONSE

Figure 14. Status Request

The contractor will reply with a bid, having in its node abstraction, his current status.

To: MANAGING NODE

From: CONTRACTING NODE

Type: BID

Contract: CURRENT CONTRACT NUMBER

Node Abstraction:

STATUS: BUSY

Figure 15. Status Reply

Normal protocol requires an award message, so the award message will be modified to be a status acknowledgement.

To: CU

From: MN

Type: AWARD

Contract: CURRENT CONTRACT NUMBER

Task Specification:

STATUS ACKNOWLEDGEMENT

Figure 16. Acknowledgement

It would not be possible to insure that the extra messages would only be issued when a node has failed, but heuristics could be devised and tuned to keep the occurence of them to some tolerable level. A tolerable level of ten percent is assumed in subsequent analysis.

The cost of this approach is $K^n(TA+B+A+R_p)$ for normal operation plus $.1K^n(TA+B+A)$ for the exchange of messages due to a status request plus $SK^{n-L}(TA+B+A+R_p)$ for the recontracting of all tasks below the failed node plus $C(x)$, the cost of computation, plus $K^{-L}C(x)$ for the additional computation of redoing tasks. S is the number of failed nodes and L is the level at which the node fails.

If n is large and L is small, the cost of redoing the tasks is significant. This encourages finding a method that allows for their recovery.

The next approach is to extend the previous one, so that, instead of simple status reports containing only the health of the contractor, interim status reports are returned, which contain outstanding subcontracts, known data, and the means of utilizing them. The manager, when he detects a failed node, would use these interim status reports to recontract the subcontractors and thus recover work already in progress.

The status request message would be the same as the previous example. What will change is the meaning of the status of a node.

In the bid format the status reply would now have in

the node abstraction a status report containing the pertinent contracts, or data, and the utilization of them.

To: MANAGING NODE

From: CONTRACT NODE

Type: BID

Contract: CURRENT CONTRACT NUMBER

Node Abstraction:

STATUS: PERTINENT CONTRACTS

A: DATA

B: (SUBCONTRACTING NODE, CONTRACT NUMBER)

C: ...

UTILIZATION

TASK-TYPE-NAME 1(A, TASK-TYPE-NAME 2(B, C))

Figure 17. Status Report

The award message would be acknowledgement that the status report had been received. The complexity of the message has increased, but, with standardized task-type-names, the necessary information should be minimal.

There are three issues to be considered at this point.

(1) When should status reports be issued?

(2) How will the subcontractors be contacted and

recovered?

(3) What happens when a subcontractor reports between the last status report and the nodal failure?

Peports should be issued upon request, but this does not, in itself, fulfill the needs of the manager. If a node fails between the award and the time the manager gets suspicious, then nothing has been gained. The task will still have to be reannounced by the manager. To correct this, an interim status report will be automatically required when a contractor has established his own subcontracts. This interim status report will contain the same information that is contained in the requested status reports.

When a failed node is detected by a manager, it goes to the most current status report, if one exists, from that node. He issues a task announcement similar to the one originally issued to the failed node. The only difference will be a new contract number. The nodes will bid on it, as before, and the manager will select the one it feels is most appropriate. When the award is given, the information contained in the status report will be included in the task specification. The new contractor will use this information to establish contact with the subcontractors. If a pertinent contract is data, then no new contract is needed. If a pertinent contract is an outstanding contract, then the new manager issues a task announcement directed to the subcontractor, requesting that the old contract be

recontracted with it.

To: SUBCONTRACTOR

From: NEW CONTRACTOR

Type: TASK ANNOUNCEMENT

Contract: NEW CONTRACT NUMBER

Task Abstraction:

TASK TYPE: RECONTRACT

Bid Specification:

STATUS

Expiration Time:

IMMEDIATE RESPONSE

Figure 18. Recontract Announcement

The subcontractor replies with a bid that contains a current status report in the node abstraction.

The award message is simply an acknowledgement that it had been received.

Aproblem may arise if a subcontractor reports between the last status report and the failure of its manager. Once the node has reported, it forgets the contract. When the

new contractor contacts the subcontracting node, it has [illegible] To prevent this from happening, it will be required that all nodes keep a record of its contracts and results, for a period of time. This should not be too much of a burden on the node, since the amount of time it would be required to maintain the record would be relatively short, and memory is relatively inexpensive. The status report it gives to the contractor would be the results.

Figure 19. Recovery from Failed Node

The complete process is illustrated in Figure 19. N_1 contracts a task with N_2, who in turn, contracts with N_3 and N_4. Once N_2 has established its subcontracts, it sends an interim status report to N_1, by means of an interim Report. When N_1 discovers N_2 has failed, it reissues the task and contracts with N_5. N_5, finding nonempty utilization and pertinent contracts clauses in

the task specification, recontracts with N_3 and N_4. It issues an inter[illegible] $_1$. [illegible]ration is [illegible], and the nodes continue to function in a normal manner. The new contractor, instead of redoing the task itself, simply waits for the reports from N_3 and N_4, and uses the indicated utilization of them, in its report.

The cost of this approach would be:

$K^n(TA+B+A+R_p+ISR)$ for normal operation, plus $.1K^n(TA+ISR+A)$ for status reports, plus $S(K+1)(TA+B+A+R_p+ISR)$ for recovery from failure plus $C(x)$.

ISR is the cost associated with the interim status reports.

The cost of recovery is no longer dependent upon the nodes' location in the tree.

There is one more special case which needs clarification. It arises when the interim status report for a node is lost with the failing node. This can happen either when the manager and the contractor fail at the same time, or when a node elects to work on a task it announced itself, and then fails.

The new contractor will be unable to get the results it expects to use in the utilization clause. There are two choices for recovery. The new contractor can terminate all subcontracts in the interim status report it received, and do the task using its own methods, or it can reconstruct the subcontract and reannounce it.

The present interim status reports only contains the subcontract number and the node which made it. Allowing

different nodes to have different capabilities makes it unlikely that the subcontract can be reconstructed from the utilization clause.

When the new manager doesn't receive a bid from the directed task announcement to recontract, it terminates the subcontracts in the interim status reports, and attempts to complete the task, using the operator that was the basis for it making a bid on the task, in the first place.

The last method is an extension of the previous one. The idea of periodic interim reports is expanded to continuous reporting. Communication links are established between the manager and the contractor. The intention is that all managers will know, all the time, exactly what is the status of their respective contractors. This would be fulfilled by extending each nodes communication capability (possibly by adding another processor to each node, whose sole purpous is to control and coordinate communication).

The contract between two nodes would be established in the normal manner. The manager would issue a task announcement, the nodes would bid on the task and the manager would award the contract to the node it prefers.

Once the contract had been made, a continuous communication link would be established between the manager and the contractor. This would enable the manager to know at all times the health of the node, the stage of task completion, and all pertinent information, such as partial results and subcontractors. When a failure occurs the

manager knows about it immediately. It reannounces the task with the award message containing data, subcontracts and utilization, as was discussed in the previous approach. The new contractor contacts the subcontractors by means of a task announcement, and has the contract remade with itself as the recipient of further communication. The communication links between the old contractor and the manager and subcontractors are broken and reestablished, in turn, with the new contractor. The cost of this system is $K^n(TA+B+A+R_p)+C(COM)+C(x)$ for normal operation, plus $S(K+1)(TA+B+A)$ for recovery from failure.

METHOD	NORMAL OPERATIONAL COSTS	RECOVERY COSTS
R^{th}-order redundancy I	$R^nK^n(TA+B+A+R_p)+R^n C(x)$	0
Periodic status requests coupled with reannouncing the task. II	$1.1K^n(TA+B+A+R_p)+C(x)$	$S(K^{n-L}(TA+B+A+R_p)+K^{-L}C(x))$
Periodic status requests with interim status reports III	$1.1K^n(TA+B+A+R_p+ISR)-.1K^n(B+R_p)+C(x)$	$S(K+1)(TA+B+A+ISR)$
Continuous communication between manager and contractor IV	$K^n(TA+B+A+R_p)+C(x)+C(COM)$	$S(K+1)(TA+B+A)$

R= order of redundancy; C(COM)= cost of communication links

K= number of decendents per node; C(x)= cost of computation

n= depth of goal tree; L=tree depth at which failure occurs

TA= cost of task announcement message; S=number of failures

B= cost of bid message; ISR= cost of interim status report

A= cost of award message; R_p= cost of report message

Figure 20. Cost of Recovery Methods

Normal operation costs are continuous but recovery costs are random and unpredictable, therefore it makes sense to break the costs of the four approaches, into these areas.

The ordering of the four methods, with respect to normal operational costs, from least expensive to most expensive, is II,III,I,IV. The cost ISR is on the order of a bid, therefore, for R greater than 1, I is more expensive than either I or II. IV, because of its communication links, is considered the most expensive of them all.

Ordering, with respect to recovery cost, yields I,IV, III,II.

Method I, the R^{th} order redundancy, is not considered to be a good solution to the problem of node failure. It is, at its best, only a partial fix. In the event of a catastrophe, redundancy will only allow survival if one node for each task is undamaged. Should all R nodes fail for a particular task, the system will fail, since it has no means of recovery. It does provide a capability for discovering eroneous reports, but this is a different, although related, problem from the one discussed in this paper.

Method IV, continuous communication between nodes, is quite expensive, due to extreme expansion of communication requirements. It has some value where the tasks assigned to the nodes are of such high priority that they outweigh the cost of the communication links. This method may require the nodes to be located near each other and

limits the number of nodes that the net can absorb.

Most applications would call for either Method II, or Method III. Method II, simple periodic status, requests can be more easily implemented than III, and has the lowest normal operational overhead. Unfortunately, it has the highest recovery costs. These costs are not independent of the location in the goal tree of the node that fails. It also loses the benefit of any subcontractors below the failed node, and forces the work to be redone. It is, therefore, best suited for applications where there are a sufficient number of nodes available so that the wasted effort is not significant, and where the tasks are of a low enough priority that the time lost for recomputation is not important.

Method III, periodic status requests with interim status reports, is suited for most applications between these poles. The cost of an interim status report is roughly the same as any other message, thus the normal operational costs of Method III are within a constant factor of Method II. Its recovery time is fast and independent of the location of the failure and it allows the retrieval of subcontractors below the failed node. Considering its cost verses effectiveness, Method III is the best, for general use, of the four methods discussed.

7. EXAMPLE OF RECOVERY

The example, in section 4, will be extended to allow recovery. The method to be used is periodic status requests and interim status reports (Method III).

The awarding of contract 1-3 by Node 1 to Node 2 is accomplished exactly as before. The task announcement, bid, and award are unchanged. Node 2 awards contracts 2-1, 3-3, and others to Node 3, Node 4, and others with the messages already given.

Once these contracts are secured, Node 2 is ready to give an interim status report.

To: NODE 1

From: NODE 2

Type: INTERIM REPORT

Contract: 1-3

Result Description:

STATUS: PERTINENT CONTRACTS:

A: (NODE 3, 2-1)

B: push (Box 2, c, b)

C: (NODE 4, 3-3)

D: other node working on push (Box 1, m, d)

UTILIZATION:

(A, B, C) $\vee$ (A, B, D) $\vee$ (A, B, E)

If Node 2 had failed before the interim status report, Node 1 would have simply reannounced the task. The status report gives Node 1 the information it needs to recover lower nodes.

Node 2 fails. Node 1 waits the appropriate period of time then it issues a status request.

To: NODE 2

From: NODE 1

Type: TASK ANNOUNCEMENT

Contract: 1-3

Task Abstraction:

TASK TYPE: STATUS REQUEST

Bid Specification:

STATUS

Expiration Time:

IMMEDIATE RESPONSE

Now Node 1 issues the same task announcement it made, for the contract with Node 2, except the contract number is different. Node 6 makes a bid based on its relevant instantiation push (Box 2, m, b). Node 1 awards the contract to it, including in the award message the information in the interim status report from Node 2.

To: NODE 6

From: NODE 1

Type: AWARD

Contract: 6-1

Task Specification:

push (Box 2, m, b)

((ATR (a) AT (Box 1, b)

AT (Box 2, c) AT (Box 3, d),

((AT (Box 1, x)

AT (Box 2, x)

AT (Box 3, x)))

PERTINENT CONTRACTS:

A: (NODE 3, 2-1)

B: push (Box 2, c, b)

C: (NODE 4, 3-3)

D: other node working on push (Box 1, m, d)

E: other node working on push (Box 2, m, d)

UTILIZATION:

(A, B, C) $\vee$ (A, B, D) $\vee$ (A, B, E)

Node 6 must establish contracts with those nodes in the task abstraction. For Example, the message to Node 3.

To: NODE 3

From: NODE 6

Type: TASK ANNOUNCEMENT

Contract: 7

Task Abstraction:

TASK TYPE: [illegible]

Bid Specification:

STATUS

Expiration Time:

IMMEDIATE RESPONSE

Node 3, Node 4, and the others reply with bids containing their current status in their node abstractions.

After Node 6 acknowledges these recontractings, sends a new interim status report to Node 1, the net continues its normal operation.

8. SUMMARY

Interest in distributed processing has been growing due to the availability of increasingly powerful microprocessors. The contract net has been proposed as a method for achieving the control and coordination required for it.

The issue of catastrophic node failure must be resolved for the contract net to be practical.

Four methods for recovery from a failure have been discussed.

(1) R^{th} order redundancy. The manager contracts with R nodes to complete the task.

(2) Simple periodic status requests. The manager checks a contractor after a period of time. If it has failed, the task is reannounced.

(3) Period status requests with interim status reports. The manager gets information from status reports which allows it to recover nodes below the failure.

(4) Continuous communication links. The manager receives continuous updates on the condition and progress of the contractor.

The first doesn't solve the problem. The fourth is too expensive, except for small nets with high priority tasks. The second is inexpensive and easily implemented, but it cannot recover nodes below the failure.

The third, periodic status requests with interim status

reports, because it is inexpensive and effective, is considered the best for general use.

9. HISTORY OF PROBLEM

There have been many attempts at planning with varying levels of success. GPS[5], STRIPS, ABSTRIPS[6] and NOAH[7] were studied for suitability as a medium for examining the issue of node failure. Although NOAH had been distributed in other work[8], STRIPS was chosen for this paper. Distributing it is sufficiently involved to bring out the issues, yet simple enough to generate understandable examples.

The author of this paper knows no other literature on the problem of node failure.

10. SUGGESTIONS FOR FUTURE WORK

Distributing STRIPS has brought out some issues which were not dealt within this paper.

The suitability of STRIPS for distribution is still quetionable. Implementing the distribution of alternative choices towards reducing a difference, is reasonable and clean, but the distribution, generated by breaking a task into subtasks, before and after operator application, was done only by deemphasizing the importance of the delete list. The trick employed may not be valid for all applications of STRIPS.

Prior to distribution of alternate choices is the problem of selection. This paper simply selected them all, ignoring both finite number of nodes in the net, and possible wide diparity between alternatives. Examining all alternatives could lead to super-saturating the net. It may be worthwhile to use only a fraction of the alternatives, the fraction,determined by the complexity of the better alternatives, or at least holding off investigating an alternative that is thought to be very unlikely. Nodes, with more than one operator instantiation, pose a problem for the manager, that might be better handled by the contractor.

The solution to the problem of failed nodes,asserted by this paper, requires that work be done in developing heuristics for determining tolerable waiting periods. Waiting too long results in excessive lost time, and waiting

too short results in excessive message exchanges. Although, it might not be possible, what would be preferred is heuristics that are independent of the class of tasks given to the nodes.

There are problems related to node failure which have not been treated at all. Lost or garbled messages might be handled by the inclusion of a repeat message. After reception of a garbled message, or an excessive wait for a message, a node could ask for the message to be repeated. For the class of problems, where checking a solution is easier than finding the solution, erroneous data from a node could be detected by requiring the manager to check the solution given it, against the task it contracted out.

11. FOOTNOTES

1. See reference 2.
2. See reference 5.
3. See pgs. 200-203 in reference 2.
4. This method was suggested in reference 5.
5. See reference 6.
6. See reference 3.
7. See reference 5.
8. See reference 1.

12. REFERENCES

1. D. D. Corkill
   "Hierarchical Planning in a Distributed Environment"
2. R. E. Fikes and N. J. Nilsson
   "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving"
   Artificial Intelligence 2 (1971), 189-208
   North-Holland Publishing Company
3. E. D. Sacerdoti
   "Planning in a Hierarchy of Abstraction Spaces"
   Artificial Intelligence 5 (1974), 115-135
   North-Holland Publishing Company
4. E. D. Sacerdoti
   A Structure for Plans and Behavior (1975)
5. R. G. Smith and R. Davis
   "Distributed Problem Solving: Transfer of Control" (1980)
6. A. Newell and H. A. Simon
   "GPS, A Program that Simulates Human Thought"
   In Computers and Thought, 279-293

DATE
FILMED

8